DOCUMENT RESUME

ED 193 026 SE 032 867

AUTHOR Stolarz, Theodore J.
TITLE The Programmable Calculator in the Classroom.
SPONS AGENCY National Science Foundation, Washington, D.C.
PUB DATE 23 Mar 79
NOTE 9p.: Contains occasional broken type.

EDRS PRICE MF01/PC01 Plus Postage.
DESCRIPTORS *Algorithms: *Calculators: Cognitive Processes: Computer Programs: Educational Technology: *Elementary Secondary Education: Instructional Materials: Learning Activities: *Mathematical Applications: Mathematical Concepts: *Mathematical Logic: *Mathematics Education: Mathematics Materials: Problem Solving
IDENTIFIERS *Programable Calculators

ABSTRACT

The uses of programable calculators in the mathematics classroom are presented. A discussion of the "microelectronics revolution" that has brought programable calculators into our society is also included. Pointed out is that the logical or mental processes used to program the programable calculator are identical to those used to program any computer. A list and description of thirteen mathematical- and computer-related concepts that students can learn by working with programable calculators is presented. The report concludes with four additional uses of these electronic devices by teachers and pupils in the classroom. (MP)

****
* Reproductions supplied by EDRS are the best that can be made *
* from the original document. *
****

U.S. DEPARTMENT OF HEALTH, EDUCATION & WELFARE
NATIONAL INSTITUTE OF EDUCATION

THIS DOCUMENT HAS BEEN REPRODUCED EXACTLY AS RECEIVED FROM THE PERSON OR ORGANIZATION ORIGINATING IT. POINTS OF VIEW OR OPINIONS STATED DO NOT NECESSARILY REPRESENT OFFICIAL NATIONAL INSTITUTE OF EDUCATION POSITION OR POLICY

"PERMISSION TO REPRODUCE THIS MATERIAL HAS BEEN GRANTED BY
Mary L. Charles of the NSF.
TO THE EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)."

THE PROGRAMMABLE CALCULATOR IN THE CLASSROOM

Theodore J. Stolarz
March 23, 1979

ED193026

INTRODUCTION I was an elementary school student in the late 1930's and attended the Ernest Prussing Elementary School in Chicago's far northwest side. It was a rather new school then. It had an excellent reputation for achievement and, as I recall, it was provided with the best of everything that could be provided during the great depression. We had one of the first "adjustment teachers" in Chicago and I remember taking a lot of tests that ultimately resulted in my skipping a grade. But what I noticed about the adjustment teacher (apart from the fact that she was an attractive young woman) was that she used a stopwatch and a slide rule. It was the first time in my life that I had seen such things. I remember that I had to build up my courage to ask her about them and she demonstrated them to me. I had no use for a stopwatch, but a ruler that could do arithmetic seemed like a miracle. I had to have one.

I saved nickels and dimes for what seemed an eternity and finally bought a wooden slide rule with "painted on" numbers and a little instruction book. The slide stuck in the rule like glue on damp summer days and fell out if you tilted it during the dry days of winter. I remember how disappointed I was when I found out that it could not add or subtract. But I was the only student in the 8th grade that had one and I carried it with me wherever I went.

Later on, at Carl Schurz High School an algebra teacher caught me using my slide rule on an examination. Her scolding made me feel as if I were "cheating." When I later found time to explain to her my reasons for using the rule she stated that it was a "crutch" and that I was doomed to going through the rest of my life carrying a slide rule. I had a nightmare where someone stole my rule and I was unable to multiply 3 by 4. The truth is, however, that she was partially correct. I have always had one with me until I purchased my Texas Instruments SR-52 calculator. My expensive Pickett slide rule now sits in my desk drawer at the college gathering dust.

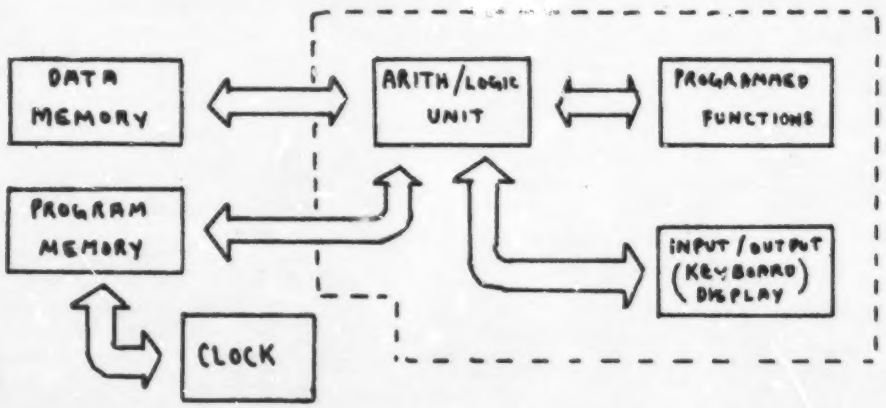
Many educators, with good reasons, fear the use of electronic calculators by children in the elementary and secondary schools. I am sure that many of you have heard their arguments. They are often difficult to answer. Last year I was grading a problem one of my students handed in for a statistics course I was teaching. About half way through the problem her numbers started making no sense at all. I couldn't follow her logic. When I asked her to explain the logic of her solution she stared at the paper for a while and then said, "I think my batteries were getting weak." I confess to you that I did not know where to begin to help her. I was also worried that a student might some day flunk out of college because a transistor failed.

SE 032 867

THE MICROELECTRONICS REVOLUTION But in reality, the "microelectronics revolution" is with us and the world will never be the same as it was before it started. It is by no means over. Whereas, in the past a teenager had to be able to add and make change to get a job we now find that in the "fast-food" hamburger shops the youngsters press buttons with pictures of hamburgers, milkshakes, and fried potatoes on them and the microprocessor in the register totals up the bill and indicates the correct change for the bills you offer in payment. Within a decade or less most college students and many high school students will have computers of their own with higher level languages such as BASIC or PASCAL and graphics capabilities. The calculator is now virtually taken for granted by children and versions of Texas Instruments "Speak and Spell" will be teaching vocabulary and spelling. The question is not, "Will we make use of calculators in the schools?," it is, "How will we do so?" In 1971 I purchased one of the first "hand held" calculators (the Monroe Model 10) for $ 350.00. A much better machine can be purchased at Radio Shack or K-Mart for about ten dollars. I have in my home a personal computer system (the Heath H-8) which is more powerful than the "MANIAC" computer which was used to verify the mathematical equations for the first hydrogen bomb which was designed at Los Alamos, New Mexico two decades ago. There is no turning the world back.

THE PROGRAMMABLE CALCULATOR We are concerned here, however, not with calculators and their uses, but with programmable calculators. It is very important that we draw a clear distinction between these two because, despite their obvious similarities, they are really very different machines. A calculator simply performs operations that are familiar to the user and can be performed by the user without the calculator. The programmable calculator introduces a new mental process and a set of new concepts that must be learned. The programmable calculator is a computer disguised as a calculator. The "stored program" concept and the use of sequential steps following an "algorithm" to solve a problem is a computer process. This difference provides us with the opportunity to develop in our students a method of thinking that will prove to be of great value as they face the world of the future.

Let us examine this difference by reference to the diagram below.



A calculator contains the features shown inside the dashed lines shown above. All actual calculations are performed by the arithmetic unit which sends the results to a display (light emitting diodes or paper tape). Input of numbers (data) is accomplished using the keyboard. Certain keystrokes require the arithmetic unit to address pre-programmed subroutines which extract square roots, calculate logarithms or trigonometric functions, or in more complex machines convert degrees to radians, solve rectangular to polar coordinates, etc.. The analogy to the central processing unit, input/output functions, and subroutine branches is obvious to anyone with a passing acquaintance with computer architecture. The similarity is even greater when we recall the fact that all operations are actually done in binary code using such standard computer techniques as two's complement arithmetic and the rules of Boolean Algebra. All that need be added to build a complete "digital computer" is to provide a program memory which is sequenced by a "clock" which feeds the steps to the arithmetic unit at a speed within the limits of its capacity and a data memory which can store numerical data on a "destructive read-in, non-destructive read-out" basis. All programmable calculators have these features and they differ only in memory sizes, speed, provision of remote recorded program entry, and the "richness" of the instruction set provided to manipulate the various "blocks" shown in the diagram.

The key point is this. The logic or "mental process" used to program the programmable calculator is identical to that used to program any computer. In the calculator, programming is done by defining a series of "keystrokes" that will solve a problem for the user. In computer languages such as "assembler languages" or so-called "higher level" languages such as ALGOL, BASIC, APL, FORTRAN, COBOL, FOCAL, PASCAL, etc. we use a fixed set of written instructions in a rigidly defined syntax to solve the problem. The mental process used by the programmer is fundamentally the same. It involves developing an "algorithm" which breaks down the solution of the problem into a series of sequential steps which must be executed in an exactly defined sequence. There is no room for ambiguity in either case.

PLANNING ALGORITHMS The key to developing programs that will solve problems using computers or programmable calculators is skill in developing algorithms. The programmable calculator is ideal for introducing this skill because the user can concentrate on the problem of how to accomplish a goal without the added confusion of learning the syntax of a computer language. This is not immediately obvious. I had the unfortunate experience of doing all this backwards. My first efforts at building algorithms were with the IBM 1401 computer which had a FORTRAN compiler. I had little prior experience with computers and no instruction in their use. I acquired the IBM technical manuals and spent some weeks before I could get any idea of how to get numbers into the card reader and results out on the printer. During most of this time I had to operate the computer myself and the sheer size of the thing was unnerving. The few programmers around the computer did not know FORTRAN and kept trying to correct my syntax, not my strategy. Most of my errors were in program flow, not syntax. The compiler caught most of my syntax errors. The computer was faithfully doing what I told it to do, not what I wanted it to do.

After about a year of this experience, I was writing many successful programs, but my efforts tended to waste a good deal of computer time and printer paper. At about this time we acquired an early programmable calculator in the psychology department. It was made by the Monroe company and we had to punch octal codes into blue cards to represent keystrokes, which was somewhat confusing at first. But I found that solving problems with this calculator gave me much greater skill in writing FORTRAN programs for the large machine. No one prior to this took the time to teach me the fundamental process of designing efficient algorithms. I was too worried about losing control of the high speed printer which could empty most of a box of paper before the computer operator could stop it or in counting card columns to construct input format statements.

When we see demonstrations later on in this conference of problem solutions using programmable calculators we should keep in mind that the process we are using includes a series of concepts that are necessary for the efficient planning of algorithms. The programmable calculator can be used to teach and/or illustrate these as we develop this skill in our students. The following is a list (not complete, of course) of the kind of concepts I am referring to.

1. The use of a "program." The key concept which students may not understand initially is that of using a "program" to solve a problem. The programmable calculator has in common with the digital computer the fact that using a supplied program allows the user to solve a problem which he may not know how to solve. Once a program is written it can be used many times. A program I write for test item analysis in 1971 is still widely used at our university (and in many other places) to evaluate multiple choice tests. It performs thousands of calculations using some rather sophisticated algorithms. Most users are unaware of the statistical niceties it contains. They do, however, understand the output it produces, and that is all that matters. This is the "magic" of the computer; programming languages are universal. Pre-written programs are available for all programmable calculators and they are very useful.

2. Planning sequential steps for problem solutions. While some texts start out with "flow charts" and diagrams which define an algorithm, the student should first learn a fact of life. A computer can only do one thing at a time. This is true of all computers, no matter how expensive or sophisticated. We can, by clever programming, make a machine seem to do several things at the same time. However, if we have one central processing unit through which all data must flow, the computer has the capacity of doing only one thing at a time. This is very evident when using a programmable calculator where the sequence of the keystrokes is important and the task must be reduced to a finite number of steps done in a certain order.

3. Planning the order of arithmetic operations. While I am sure that the teachers of mathematics present here today teach their students that an

algebraic expression or formula is merely a convenient "shorthand" for specifying the order of arithmetic operations, some of my students act as if they do not know this. Planning the series of keystrokes needed to solve an equation can reinforce the student's understanding of the importance of this order. For example: $a + (b/c) \neq (a + b)/c$.

4. Memory storage and retrieval. Skill in manipulating memory is a mark of the good programmer. The concept of depositing intermediate results into a memory "bucket" for later use is not intuitively obvious. Less obvious is the placement of a constant like pi or e into a "bucket" for later use on a repeated basis. This is easier to see on a programmable calculator than on a computer using a language like BASIC where we might say, "LET X2=2.712828." The beginner does not see that the computer will reserve a fixed number of binary digits in memory to hold the value of the variable designated as X2. Most programmable calculators allow the user to add to a memory location or subtract a value from its contents directly. This is excellent practice.

5. Planning input/output operations. The output of a calculator is typically a number (or a series of numbers) on a lighted display or on a paper tape. The builder of the algorithm must plan when the program is to display these numbers and he must "document" his program so the user will be able to identify what the numbers mean. He or she must also plan the program so that it will pause so that data can be entered into the sequence via the keyboard so that program flow can procede. This is a skill that has a positive transfer to the learning of computer programming in the future.

6. Branching forward or backward. While program execution is sequential the sequence does not always procede in a single direction. Some parts of a program may be executed only one time while others may be executed many times to accumulate sums, products, quotients, etc. This is easily accomplished by "labels" in many calculators. Some calculators allow the user to set "flags" which can be tested to bypass "GOTO" instructions which branch programs back over previous steps. Practice with these techniques is very valuable.

7. Conditional branching. Most programmable calculators allow tests of the value in the "accumulator" or display register to see if it is positive, negative, or zero before branching to a portion of the program occurs. The test for "greater than, less than, or equal to" a certain value is easily accomplished. Such branching is so frequently used in computer algorithms that it is actually difficult to write a program of any complexity that does not use this technique.

8. Forming repetitive "loops." Using the above concepts it is possible to teach the construction of programs which "loop" through a series of steps for either a fixed number of times or until a desired result is accomplished. While this is planned for in programmables which have a "decrement memory and branch when zero" instruction, it can be done with any programmable which

provides for the "+-0" branch test. Learning the concept of "looping" is difficult for many students who first attempt it with "FOR-TO" loops in BASIC or "DO" loops in FORTRAN. It is one of the most widely used techniques in the construction of computer algorithms.

A good exercise to teach this concept is to have the student write a program to obtain the square root of a number using the method of successive approximations suggested by Newton. This is how many computers calculate square roots.

9. Clearing storage. The fact that memory locations can hold "garbage" from previous program steps and must be "cleared" or set to zero if they are to be added to or subtracted from will become painfully clear after students write programs that "bomb out." I violated this rule the other day after over a decade of computer and calculator use and hundreds of successful programs in use all over the country.

10. Overlaying memory. It is an excellent builder of mental discipline to fit a large program into a machine that is limited to a small number of memory storage locations. Clearing and re-using memory locations builds good programming skills. I find myself sometimes getting careless in this regard when using my home computer which has 24,000 bytes of memory which will soon be expanded to 40,000 bytes. Starting with a large machine can develop bad habits.

11. Use of subroutines. Most programmables allow for construction of subroutines to perform repetitive chores with economy of program steps and memory storage. This is a concept that takes some practice before it becomes a matter of habit.

12. Indirect memory addressing. Some programmables allow for branching to a program step specified in a memory location. This value can be the result of calculations within the program. This is an advanced skill but it is within the capabilities of bright students. It provides a very powerful tool in the construction of some algorithms.

13. Concern for program length and execution speed. There are, of course, several ways to construct algorithms to solve any specific problem. All of them are "correct" if they produce the desired results. The "best" algorithm is the one that provides the result using the fewest program steps or keystrokes. Since the "clock" in a programmable calculator is relatively slow as compared to a computer "clock" this speed difference is more obvious when using the calculator. The shorter program is also easier to load since there is less chance for error.

The list is not complete. However, it does suggest how much a person can learn from the use of a programmable calculator. The little machine that

fits into a pocket or purse is in fact an extremely complex marvel which contains thousands of electronic components arranged so that they can be instructed to solve a myriad of problems. In at least this one area, that of teaching the building of algorithms, its use in the classroom is very valuable. The positive transfer of these learnings to future learnings should be obvious.

OTHER USES While up to now my interests in computer algorithms, programming languages, and digital electronics clearly have influenced my presentation of how programmable calculators can be used effectively in teaching in elementary and secondary schools, there are many other uses that can be found for them that have educational value. It is a feature of this conference to explore these potential uses through the exchange of ideas. The shared creativity of many skilled educators can open up avenues and approaches that could not be foreseen by a single person regardless how creative he or she may be. Again I must reach into my own experiences which merely suggest how fruitful the field is. I will illustrate a few ideas that I have. In most of these the teacher does the "programming" but there is no reason why a student or a group of students could not be involved in this effort.

1. Curve plotting. In teaching the concept of the "normal curve" to my introductory statistics students I try to convey the idea that the curve has no "standard" shape but is a plot of an equation involving certain constants and variables. The constants are pi and e and the variables are the mean, the number of scores, the standard deviation, and the individual scores in the distribution. By writing a program for the calculator which solves for the height given the values of the variables previously mentioned, I can plot the curve several times to show what effect the variables have on the shape of the curve. Solving

$$h = \frac{n}{S\sqrt{2\pi}} e^{-(X-M)^2/2S^2}$$

for h given n, S, M, and X fifty or a hundred times is a prospect no man can face with equanimity. A short program for my Texas Instruments SR-52 gives the values of h in a few seconds and allows the curve plotting to go on. In a secondary school class the "feel" for simple equations employing powers of 2 or 3 could be of value. It might also be useful to plot lines by selected points in linear methods where a program for a "least squares" fit of a line to a set of points is easily written.

2. Building tables. I have written simple programs using the formulas for compound interest to determine how much I must save, given current bank interest to have enough money to purchase additional memory or peripheral devices for my computer at a certain point in time. I also calculated depreciation schedules for the SR-52 for my income tax form. I can see possible uses for this kind of programming in teaching business mathematics or even consumer education to students who will be users of credit in a few years.

3. Science education. I received a "lunar lander game" program with the SR-52 and I spent a few hours playing with it. Most of the time I impacted the moon at various velocities and theoretically killed myself several times. I can see, however, that the acceleration of a falling body could be demonstrated, or the trajectory of an artillery shell estimated. The ease of plotting data points gives a better feel for the dynamic nature of what is happening rather than the static feel one gets when it takes considerable time to establish a single data point. I have also used the metric conversion programs when my favorite scotch suddenly appeared in half liter bottles and I wanted to calculate the price per fifth gallon.

4. Mathematics classes. The combination of the availability of trigonometric functions and programming capability helped me verify the correct operation of a "Biorythm Plotting Program" I placed on my home computer and the computer system used by our university. The sine function is used to generate the curves, of course, but verifying the plotted points in the computer program helped "debug" the operation of the program and insure that the leap years were handled properly. I can see many possible uses for this kind of programming in courses in algebra, geometry, solid geometry, trigonometry, etc.. My lack of familiarity with the secondary curriculum in these areas, of course, limits me in anticipating uses.

I will leave the rest to you and I am eager to see how much I will learn before tomorrow is over. Thank you for your kind attention and I hope that you will find this conference to be of value in our common goal of improving the mathematics skills of our students.

- End -